arXiv:comp-gas/9801002v1 17 Jan 1998

LINDENMAYER SYSTEMS AS A MODEL OF COMPUTATIONS

Y. Ozhigov

Abstract.

LS is a particular type of computational processes simulating living tissue. They use an unlimited branching process arising from the simultaneous substitutions of some words instead of letters in some initial word. This combines the properties of cellular automata and grammars. It is proved that

1) The set of languages, computed in a polynomial time on such LS that all replacing words are not empty, is exactly NP- languages.

2) The set of languages, computed in a polynomial time on arbitrary LS , contains the polynomial hierarchy.

3) The set of languages, computed in a polynomial time on a nondeterministic version of LS, strictly contains the set of languages, computed in a polynomial time on Turing Machines with a space complexity n^a, where a is positive integer.

In particular, the last two results mean that Lindenmayer systems may be even more powerful tool of computations than nondeterministic Turing Machine.

1.Introduction and definitions

The simulation of nondeterminism by a physical device remains attractive but unattainable dream. Even the most powerful physical computational device - quantum computer is not able to achieve this objective in the computations with oracles as follows from the result of C.Bennett, E.Bernstein, G.Brassard and U.Vazirani [1]. This is most likely to be true also for absolute (without oracles) computations. Therefore living things present, probably the best real model of nondeterminism in nature.

The simple mathematical model of the structures with two fundamental properties of a Life: reproductions and determined stable interactions with the nearest neighborhood, was proposed by A.Lindenmayer ([4],[5][6]). One-dimensional Lindenmayer systems considered in this paper is a crude model of a reproduction because we ignore the spatial arrangement of tissue, otherwise this model is well suited to describe the behavior of living cells. The spatial arrangement of tissue can be often ignored for example, for the filamentous organisms.

Lindenmayer system also bears a resemblance to an imaginary nondeterministic computer with the supernatural abilities. These abilities is the subject for study in this work. We proceed with an exact definition.

Let $\omega = \{a_0, a_1, \dots, a_n\}$ be our basic alphabet, ω^* denotes the set of all words from ω and $\Sigma = \{a_0 A a_0 \mid A \in \omega^*\}$ will be our basic set of words. If $x \in \omega^*$, then $|x|$ means the length of the word x. Let $\mathcal{A}$ be a function of the form:

$$\mathcal{A}:\ \omega^3 \longrightarrow \omega^*,$$

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX

where $\mathcal{A}(a_i, a_0, a_j) = a_0$ for all $i, j = 0, 1, \ldots, n$. Then $\mathcal{A}$ is called LS. If here $\forall x \in \operatorname{Im} \mathcal{A}\ |x| > 0$ then $\mathcal{A}$ is called L-system with nonempty results - LSN.

Finally, if here $\forall x \in \operatorname{Im} \mathcal{A}\ |x| = 1$, then $\mathcal{A}$ is an ordinary cellular automaton (CA) (look at [10]).

Evolutions generated by LS (L-languages) and their algorithmic properties were subject for study mainly since 1971. As a typical example of mathematical problem here we refer to an algorithmically undecidable problem of coincidence for languages, determined by two LS (look at the work of A.Lindenmayer [6]). W.J.Savitch in [8] gave some conditions which should be imposed on the nontrivial set of strings to ensure that it is L-language; this conditions use stack machines. J. van Leeuwen in [3] obtained upper bounds for space required to recognize some extensions of L-languages (by natural operations such as sum and intersection) deterministically and nondeterministically. P.Vitanyi in [9] treated context sensitive table L-languages. Other interesting examples may be found in [7].

In the present paper LS will be considered as a particular type of computations using unlimited branching processes which arise if the length of some word $\mathcal{A}(a_i a_j a_k)$ is more than 1. Such a process requires exponentional space but reduces the time for solutions of NP-complete problems. The advantage of such computational processes are their simplicity and power. The simplicity of rules for LS allows to find them by evolutionary programming as in the work [2] of J.R.Koza.

In this paper we study a polynomial complexity of computations on LS. LS may be considered as a sort of cellular automata with reproductions which resembles a nondeterminism. It is founded that the set of languages computed in a polynomial time on deterministic as well as on nondeterministic version of LSN is coincident with NP. Thus the complexity status of LSN is clear with a precision of $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$-problem. The following peculiarities of LS must be mentioned explicitly.

The first is that the possibility of an empty results $\mathcal{A}(a_i, a_j, a_k)$ means the annihilation of some domains in a computational space and so it allows to transmit an information faster comparatively with LSN. In the section 4 it is proved that the set of languages computed on LS in a polynomial time contains the polynomial hierarchy.

The second is that this presumably makes LS inconvenient for analysis and its complexity status is still obscure.

Let P_α be the set of languages, which can be computed on Turing Machines in a polynomial time with a space complexity n^α, $\alpha = 1, 2, \ldots$. The sole lower bound of complexity result having to do with LS is that the set of languages computed on the nondeterministic version of LS in polynomial time strictly includes P_α.

Remark. *All inclusions established in this paper can be relativized. This property may be traced by a reader in all constructions.*

If $\mathcal{A}$ is LS and

$$B = a_{i_0} a_{i_1} a_{i_2} \ldots a_{i_p} a_{i_{p+1}} \tag{1}$$

is an arbitrary word from Σ, $a_{i_0} = a_{i_{p+1}} = a_0$, the new word $\mathcal{A}(B)$, defined by $\mathcal{A}(B) = a_0 A_1 A_2 \ldots A_p a_0$ where $\forall j = 1, \ldots p \ \ A_j = \mathcal{A}(a_{i_{j-1}}, a_{i_j}, a_{i_{j+1}})$, is called the result of $\mathcal{A}$-action on B.

Let W be the underlined occurrence in the word B:

$$a_0 a_{i_1} \ldots a_{i_{s-1}} \underline{a_{i_s} a_{i_{s+1}} \ldots a_{i_t}} a_{i_{t+1}} \ldots a_{i_p} a_0.$$

The base of W is the word $a_{i_s}a_{i_{s+1}}\dots a_{i_t}$. Then the underlined occurrence W_1 in the word $\mathcal{A}(B)$:

$$a_0A_{i_1}\dots A_{i_{s-1}}\underline{A_{i_s}A_{i_{s+1}}\dots A_{i_t}}A_{i_{t+1}}\dots A_{i_p}a_0$$

is called the descendant of W in transformation $B \longrightarrow \mathcal{A}(B)$. If $A_j \neq a_{i_j}$ then an occurrence of a_{i_j} in B will be considered to be affected in this transformation.

A sequence

$$B_0, B_1, \dots, B_N$$

is called an evolution of $\mathcal{A}$ if $\forall i = 1, \dots, N \quad B_i = \mathcal{A}(B_{i-1})$. If $B = B_0$, the descendant of W in this evolution is defined by induction on N. This evolution is called complete if the letter a_n occurs only in the word B_N and not in $B_1, \dots, B_{N-1}$.

We say that a set M of some words in alphabet $\omega_1 = \{a_1, \dots, a_{n-1}\}$ is computed on LS $\mathcal{A}$ if for any $B \in \omega_1^*$ the following conditions are equivalent:

1) there exists a complete evolution $B_0, B_1, \dots, B_N$, where $a_0Ba_0 = B_0$.

2) $B \in M$.

In this case for $B \in M$ we denote the number N from 1) by $\tau_{\mathcal{A}}(B)$. The set M, computed on $\mathcal{A}$, is denoted by $M_{\mathcal{A}}$.

The time complexity of LS $\mathcal{A}$ is defined by the following

$$t_{\mathcal{A}}(n) = \max_{|B|\leq n,\ B\in M} \tau_{\mathcal{A}}(B).$$

Let PLS (PLSN) denote the class of languages, computed on LS (on LSN) in a polynomial time.

Now let us define all corresponding notions for a nondeterministic version of LS (LSN). To denote these notions we add the letter "N" : NLS, NLSN, etc.

For an arbitrary set C let 2^C denote the set of all subsets of C. Thus, NLS is a function of the form

$$\mathcal{A}:\ \omega^3 \longrightarrow 2^{\omega^*}.$$

If $\forall \bar{b} \in \omega^3 \quad \forall C \in \mathcal{A}(\bar{b}) \quad |C| > 0$, then $\mathcal{A}$ is called NLSN. If a word B from Σ has the form (1) then the result of $\mathcal{A}$-action on B is the set

$$\mathcal{A}(B) = \{a_0A_1A_2\dots A_pa_0 \mid \forall j = 1, \dots, p \ A_j \in \mathcal{A}(a_{i_{j-1}}, a_{i_j}, a_{i_{j+1}})\}.$$

A sequence $B_0, \dots B_N$ is called an evolution of $\mathcal{A}$ if $\forall i = 1, \dots, N \ \ B_i \in \mathcal{A}(B_{i-1})$. A complete evolution is defined as above.

If $B \in \omega^*$ let $\tau_{\mathcal{A}}(B)$ be the minimal length of complete evolutions beginning with $B_0 = a_0Ba_0$ if such an evolution exists, and $\tau_{\mathcal{A}}(B) = \infty$ in the opposite case.

NLS $\mathcal{A}$ admits a word B iff $\tau_{\mathcal{A}}(B) < \infty$. Thereafter, the time complexity of $\mathcal{A}$ can be defined by

$$t_{\mathcal{A}}(n) = \max\{\tau_{\mathcal{A}}(B) \mid |B| \leq n,\ \tau_{\mathcal{A}}(B) < \infty\}.$$

$M_{\mathcal{A}}$ denotes here the set of all words admitted by $\mathcal{A}$, we say that this set is computed by $\mathcal{A}$.

So, NPLS(N) denotes the class of all sets, computed on NLS(N) in a polynomial time. The first two results to be established in this paper are the following.

Theorem 1. PLSN = NPLSN = NP

Theorem 2. $\forall \alpha = 1, 2, \dots \quad \mathrm{NPLS} \neq \mathrm{P}_\alpha$

2. Deterministic version of LSN

Proof of Theorem 1

The deduction of inclusion

$$\mathrm{NP} \subseteq \mathrm{PLSN} \tag{2}$$

will be our initial concern. Note that the establishment of the belonging $c \in$ PLSN, where C is an arbitrary NP-complete set would be ample. For example, let SAT be the set of all satisfiable formulas of a propositional logic. The inclusion (2) reduces to the establishment of the following belonging:

$$\mathrm{SAT} \in \mathrm{PLSN}\,.$$

Let $\bar{l} = \{\&, \vee, \neg, (,), 0, 1\}$ be the basic alphabet for a propositional logic. Let any Boolean variable α_i be coded by the word $0\underbrace{11\dots1}_{i \text{ times}}0$. Then any Boolean formula ϕ can be encoded in $\bar{l}$ and let $\ulcorner\phi\urcorner$ be its code. Let $\bar{s} = \{f, t\}$ be an auxiliary alphabet for Boolean values: t - true and f - false.

Let a special alphabet ω_1 consist of the following parts:
1) the alphabet $\bar{l}$ for the propositional logic;
2) the alphabet $\bar{s}$ for the Boolean values;
3) auxiliary letters $\alpha, \beta, b, e, h, *, +, \%, \nu$;
4) alphabets of doubles: L_1, L_2, L_3, L_4, L_5 for the letters from
$L = \bar{l} \cup \{\alpha, h, b, e, \nu\} \cup \bar{s}$ (if $x \in L$ we'll denote its doubles by $x' \in L_1, \tilde{x} \in L_2, x^0 \in L_3, x^+ \in L_4, \bar{x} \in L_5$ respectively).
5) the alphabet $L_5 \times L_5$ whose elements are denoted by $\binom{d_1}{d_2}$, where $d_1, d_2 \in L_5$.

Lemma 1. *There exists LSN $\mathcal{A}_1$ with the alphabet ω_1, such that for any propositional formula ϕ (which has been encoded as it is pointed out above) there exists the following evolution of $\mathcal{A}_1$:*

$$B_0 = a_0\ulcorner\phi\urcorner a_0, B_1, \dots, B_N, \dots$$

where:
1) $N = 4|\ulcorner\phi\urcorner|^2 + 7|\ulcorner\phi\urcorner| + 15$, B_N has the form

$$a_0 e A_1 e A_2 e \dots e A_q e a_0, \tag{3}$$

$q = 2^{|\ulcorner\phi\urcorner|}$, $\forall i = 1, 2, \dots, q$ $A_i = S_i\%\ulcorner\phi\urcorner$, $S_i \in \bar{s}^*$, $|S_i| = |\ulcorner\phi\urcorner|$, and for all i, j if $1 \le i < j \le q$ then $S_i \neq S_j$.
2) $\forall j = 2, 3, \dots, N$, A, B if

$$B_j = A\%B \tag{4}$$

then $\exists A', B'$: $A = A'S'$, $B = \ulcorner\phi\urcorner B'$, where $S' \in \bar{s}^*$, $|S'| = |\ulcorner\phi\urcorner|$, and the underlined occurrence in the word B_j : $A'\underline{S'\%\ulcorner\phi\urcorner}B'$ is not affected in the evolution $B_j, B_{j+1}, \dots$.
3) $\forall j > N$ $B_j = B_N$.

Proof of Lemma 1

It is not difficult to understand how such an automaton $\mathcal{A}_1$ can be constructed. It must replicate the words of the form $Sb^k\alpha\phi$ for various values of $S \in \bar{s}^*$ sequentially, where for all time $|Sb^k| = |\ulcorner\phi\urcorner|,\ |S| = 1, 2, \dots,\ b$ is the special letter, $k \in \mathbb{N}$. And what is essential for the present purposes, this can be done in a polynomial time for the input data $\ulcorner\phi\urcorner$ by the definition of LSN.

Any LS $\mathcal{A}$ with an alphabet ω may be determined by the list of all records of the form

$$(a, b, c) \longrightarrow \mathcal{A}(a, b, c);\ a, b, c \in \omega, \mathcal{A}(a, b, c) \neq b,$$

which may be thought of as active commands. In what follows any list of commands is taken to be supplemented by all plausible (passive) commands of the form

$$(a, b, c) \longrightarrow b$$

which are not written explicitly.

The list of commands, determining $\mathcal{A}_1$ consists of two groups:

Group 1.

input: $a_0\ulcorner\phi\urcorner a_0$,
output: $a_0\underbrace{b\dots b}_{|\ulcorner\phi\urcorner|}\alpha\ulcorner\phi\urcorner e a_0$
time: $\leq 7|\ulcorner\phi\urcorner| + 5$.
x, y take all values from $\bar{l}$, z takes all values from $\bar{l}$ and its doubles from L_2, ? - from ω_1.

$$\begin{array}{ll}
(x, a_0, ?) \longrightarrow ea_0, & (?, x', \tilde{y}) \longrightarrow \tilde{x}, \\
(?, a_0, x) \longrightarrow a_0+, & (?, x', e) \longrightarrow x^0, \\
(?, +, z) \longrightarrow \beta+, & (?, x^0, ?) \longrightarrow x, \\
(?, a_0, \beta) \longrightarrow a_0 e, & (?, x', y^0) \longrightarrow x^0, \\
(+, x, ?) \longrightarrow x', & (?, +, x^0) \longrightarrow \alpha, \\
(x', y, ?) \longrightarrow \tilde{y}, & (?, \beta, \alpha) \longrightarrow b, \\
(?, \tilde{x}, ?) \longrightarrow x', & (?, \beta, b) \longrightarrow b.
\end{array}$$

Group 2.

input: $a_0 e b^q \alpha\ulcorner\phi\urcorner e a_0$,
output: (3)
time: $\leq 4|\ulcorner\phi\urcorner|^2 + 10$,
w, x, y, z, u take all values from $L - \{e\}$.

$$(v,b,?)\longrightarrow\tilde\nu\ (v\in\omega_1-\{b,\beta\}),$$
$$(?,x,\tilde y)\longrightarrow\tilde x,$$
$$(r,\tilde b,?)\longrightarrow\tilde\nu,\ (r\in\bar s\cup\{e,*,a_0\}),$$
$$(\rho,\tilde x,?)\longrightarrow\bar x,\ \rho\in\{*,e\},$$
$$(\bar x,y,?)\longrightarrow\bar y,$$
$$(\bar x,\bar y,e)\longrightarrow\begin{pmatrix}\bar y\\ \bar y\end{pmatrix}\ (x\notin\bar s\text{ or }y\neq b),$$
$$(\bar x,e,?)\longrightarrow **,$$
$$(\begin{pmatrix}\bar x\\ \bar y\end{pmatrix},*,?)\longrightarrow *y,\ (y\neq\nu),$$
$$(\begin{pmatrix}\bar x\\ \bar\nu\end{pmatrix},*,?)\longrightarrow *t,$$
$$(\bar w,\bar x,\begin{pmatrix}\bar y\\ \bar z\end{pmatrix})\longrightarrow\begin{pmatrix}\bar x\\ \bar h\end{pmatrix},$$
$$(x,\begin{pmatrix}\bar y\\ \bar z\end{pmatrix},?)\longrightarrow\begin{pmatrix}\bar y\\ \bar x\end{pmatrix},$$
$$(\begin{pmatrix}\bar x\\ \bar d\end{pmatrix},\begin{pmatrix}\bar y\\ \bar g\end{pmatrix},?)\longrightarrow\begin{pmatrix}\bar y\\ \bar d\end{pmatrix},\ d,g\in L\cup\{h\},$$
$$(c,\bar x,\begin{pmatrix}\bar y\\ \bar h\end{pmatrix})\longrightarrow x,$$
$$x\neq\nu,\ c\in\{e,*,a_0\},$$
$$(c,\bar\nu,\begin{pmatrix}\bar x\\ \bar h\end{pmatrix})\longrightarrow f,\ c\in\{e,*,a_0\},$$
$$(x,\begin{pmatrix}\bar y\\ \bar z\end{pmatrix},?)\longrightarrow y,\ x\notin\bar s\text{ or }y\neq\nu,$$
$$(r,\begin{pmatrix}\bar\nu\\ \bar x\end{pmatrix},?)\longrightarrow f,\ r\in\bar s,$$
$$(x,*,?)\longrightarrow e,$$
$$(\bar r,\bar b,e)\longrightarrow\begin{pmatrix}\bar b\\ \bar\nu\end{pmatrix},$$
$$(x,\alpha,?)\longrightarrow\%.$$

Informal comments.

Let an evolution of $\mathcal{A}_1$ begin with the word $a_0\ulcorner\phi\urcorner a_0$.

1) An occurrence of word xb where $x\notin\{b,\beta\}$ originally arises in the output of Group 1: $a_0\ulcorner\phi\urcorner a_0$.

2) The first active command from Group 2 operating in the evolution in question is $(e,b,?)\longrightarrow\tilde\nu$, so the input of Group 2 has the form $a_0\ulcorner\phi\urcorner a_0$.

Let ϕ contain Boolean variables $\alpha_1,\alpha_2,\dots,\alpha_q$.

Lemma 2. *There exists CA $\mathcal{A}_2$ such that 1) For any active command of $\mathcal{A}_2$: $(x,y,z)\longrightarrow w\quad y,w\notin\omega_1$ and if $t\in\{x,z\},t\in\omega_1$, then $t=\%$. 2) For any input data of the form (4) from the statement of Lemma 1 the following conditions are equivalent:*

A). There exists a complete evolution of $\mathcal{A}_2$ of the form

$$B_j=B'_j,B'_{j+1},\dots,B'_{N'},\tag{5}$$

B). B_j contains an occurrence of the word $S\%\phi C$, where the end of the length q of the word S constitutes the list of values for Boolean variables which makes ϕ true. 3) Any complete evolution of $\mathcal{A}_2$ of the form (5) has the length $N'=O(|\ulcorner\phi\urcorner|^2)$.

Proof of Lemma 2

$\mathcal{A}_2$ realizes a conventional procedure for the search of true value for ϕ. Such a procedure is well known and it requires a quadratic time on Turing Machines. Lemma 2 is proved.

Returning to the proof of inequality (2), note that if the list of active commands for LSN $\mathcal{A}$ is obtained from such lists for $\mathcal{A}_1$ and $\mathcal{A}_2$ by a simple join of them, then $\mathcal{A}$ recognizes the satisfability of Boolean formulae in a quadratic time that implies inequality (2).

It is evident that PLSN $\subseteq$ NPLSN. So, the establishment of the following inclusion

$$\text{NPLSN} \subseteq \text{NP} \tag{6}$$

would suffice to accomplish the proof of Theorem 1. We shall take up this subject now.

Given NLSN $\mathcal{A}$ computing the set $M_{\mathcal{A}}$ in a polynomial time, one need only to obtain a nondeterministic cellular automaton $\mathcal{A}^*$, which computes $M_{\mathcal{A}}$ in a polynomial time.

Let

$$B_1, B_2, \ldots, B_N \tag{7}$$

be an evolution of NLSN $\mathcal{A}$ and let W_i be some selected and underlined occurrence in B_i : $B_i = B'_i\underline{A'_i a_{j_i} A''_i}B''_i$ for each $i = 1, 2, \ldots, N$, so that

1) for all $i = 1, \ldots, N$

if $B'_i \neq \Lambda$, then $|A'_i| = N + 1$,

if $B''_i \neq \Lambda$, then $|A''_i| = N + 1$, where Λ denotes the empty word,

2) for all $i = 1, \ldots, N - 1$ W_{i+1} is contained in the descendant of W_i in the transformation $B_i \longrightarrow B_{i+1}$,

3) if U_i is separated occurrence of a_{j_i} in W_i then U_{i+1} is contained in the descendant of U_i in the transformation $B_i \longrightarrow B_{i+1}$.

In this case the sequence of occurrences

$$W_1, W_2, \ldots W_N \tag{8}$$

is called a local sequence for the evolution (7), and the sequence

$$a_{j_i}, a_{j_2}, \ldots a_{j_N} \tag{9}$$

is called the central sequence for evolution (7) corresponding to the local sequence (8).

The following property of NLSN follows immediately from the definitions.

Property of localization. *Let the sequence (9) be the central sequence for evolution (7) corresponding to the local sequence (8). Then for any $i \in \{1, \ldots, N\}$ and for any words C'_i, C''_i the sequence $a_{j_i}, a_{j_{i+1}}, \ldots, a_{j_N}$ will be a central sequence for evolution $B'_i, B'_{i+1}, \ldots, B'_N$, where $B'_i = C'_i A_i a_{j_i} A''_i C''_i$.*

This property makes it possible to simulate NLSN $\mathcal{A}$ on a nondeterministic cellular automaton $\mathcal{A}^*$. The work of $\mathcal{A}^*$ consists of sequential cycles, each of them has an input I_i and an output I_{i+1} with separated occurrences.

Let $I_i = a_0 A'_i \underline{a_{j_i}} A''_i a_0$ be the input of some step number i with the separated occurrence, and $a_0 B' \underline{A} B'' a_0 \in \mathcal{A}(I_i)$ where $\underline{A}$ is the descendent of $\underline{a_{j_i}}$. Let $A = A_1 a_{j_{i+1}} A_2$ be some representation of A.

Let $t_{\mathcal{A}}(n)=O(n^p),\ p\in\mathbb{N},\ M=|I_1|$, where I_1 is an input of $\mathcal{A}$.

Let A'_{i+1} be the maximal from the ends of $B'A$, whose lengths do not exceed $t_{\mathcal{A}}(M)+1$.

Let A''_{i+1} be the maximal from the beginnings of A_2B'', whose lengths do not exceed $t_{\mathcal{A}}(M)+1$. Then

$$I_{i+1}=a_0A'_{i+1}\underline{a_{j_{i+1}}}A''_{i+1}a_0.$$

The automaton $\mathcal{A}^*$ described computes the set $M_{\mathcal{A}}$ due to Property of localization. It is readily seen that $\mathcal{A}^*$ requires the time $O(n^{2p})$. Theorem 1 is proved.

Note that Theorem 1 gives PLS $\subseteq EXPTIME$ in particular.

3. Nondeterministic version of LSN

Theorem 2: Sketch of the proof.

A conventional procedure of diagonalization will be run by NLS. Namely, if $\ulcorner\mathcal{B}\urcorner$ denotes a code of CA $\mathcal{B}$, we shall construct such NLS $\mathcal{A}$ with the time complexity $t_{\mathcal{A}}(n)=O(n)$ that for any alphabet ω' and for any $\alpha=1,2,\dots$ there exist $c_1,c_2>0$ so that for any CA $\mathcal{B}$ in alphabet ω' the following two conditions are equivalent.

1). There exists a complete evolution of $\mathcal{B}$ of the form

$$B'_0=a_0\ulcorner\mathcal{B}\urcorner a_0,B'_1,\dots,B'_M$$

where $m=|\ulcorner\mathcal{B}\urcorner|, M\le exp(c_1m),\ \forall i=1,\dots,M\ |B'_M|\le m^\alpha$ and B_M contains a_{n-1}.

2). There exists a complete evolution of $\mathcal{A}$ of the form

$$B_0=a_0\ulcorner\mathcal{B}\urcorner a_0,B_1,\dots,B_h,\ h\le c_2m^\alpha.$$

I drop all details.

4. LS and the polynomial hierarchy

There is an intimate connection between LS and the well-known polynomial hierarchy (PH).

PH is determined by the sequences of classes:

$$\Sigma\mathrm{P}_0\subseteq\Sigma\mathrm{P}_1\subseteq\dots,$$
$$\Pi\mathrm{P}_0\subseteq\Pi\mathrm{P}_1\subseteq\dots,$$

defined by the following induction.

Basis. $\Sigma\mathrm{P}_0=\Pi\mathrm{P}_0=\mathrm{P}$ - is the class of all predicates, computed in a polynomial time on Turing Machines.

Step. 1. Let $\Sigma\mathrm{P}_n$ be the class of all predicates $A(x_1,\dots,x_s)$ such that

$$A(x_1,\dots,x_s)\Longleftrightarrow\exists x_{s+1}:\ |x_{s+1}|\le p(|x_1|,\dots|x_s|)\ \ B(x_1,\dots,x_s,x_{s+1})$$

for some polynomial p and some $B\in\Pi\mathrm{P}_{n-1}$.

2. Let $\Pi\mathrm{P}_n$ be the class of all predicates $A(x_1,\dots,x_s)$ such that

$$A(x_1,\dots,x_s)\Longleftrightarrow\forall x_{s+1}:\ |x_{s+1}|\le p(|x_1|,\dots|x_s|)\ \ B(x_1,\dots,x_s,x_{s+1})$$

for some polynomial p and some $B\in\Sigma\mathrm{P}_{n-1}$.

Then $\mathrm{PH} = \bigcup_{n=0}^{\infty} \Sigma\mathrm{P}_n = \bigcup_{n=0}^{\infty} \Pi\mathrm{P}_n$.

Adding the sign of coma to alphabet ω we naturally extend the class PLS of predicates to predicates $A(x_1, \ldots, x_n)$ on $\underbrace{\omega^* \times \cdots \times \omega^*}_{n \text{ times}}$ for any $n = 1, 2, \ldots$.

Theorem 3. $\mathrm{PH} \subseteq \mathrm{PLS}$.

PROOF

It is sufficient to prove that for all $n = 0, 1, \ldots$ the following two inclusions take place:

$$\Sigma\mathrm{P}_n \subseteq \mathrm{PLS} \tag{10}$$

$$\Pi\mathrm{P}_n \subseteq \mathrm{PLS}\,. \tag{11}$$

This will be proved by a simultaneous induction on n.

Basis Follows from Theorem 1.

Step

1) Let us prove the inclusion (10).

Given LS $\mathcal{B}$ computing a predicate $B(x_1, \ldots, x_s, x_{s+1}$ in a polynomial time p_1 by the inductive hypothesis, the LS $\mathcal{A}$ must be constructed computing in a polynomial time the predicate

$$A(x_1, \ldots, x_s) \Longleftrightarrow \forall x_{s+1} : \ |x_{s+1}| \leq p(|x_1|, \ldots |x_s|).$$

We can suppose that p and p_1 are increasing positive functions.

It will be readily seen how to define such LS $\mathcal{A}$ with the help of new auxiliary letters in the following sequential steps.

Step 1 *Input* :

$$x_1, x_2, \ldots, x_s$$

Output

$$\alpha\tilde{x}_1, \ldots, \tilde{x}_s, \tilde{x}_{s+1}\%\tilde{x}_1, \ldots, \tilde{x}_s, \tilde{x}_{s+2}\% \ldots \%\tilde{x}_1, \ldots, \tilde{x}_s, \tilde{x}_{s+M}\alpha, \tag{12}$$

where $M \leq n^{p(|x_1|, \ldots, |x_s|)+1}+1$, $x_{s+1}, \ldots, x_M$ are all the different words of the length $\leq p(|x_1|, \ldots, |x_s|)$, $x_i = y_{i1}y_{i2} \ldots y_{is_i}$, all y_{ij} are letters, $\tilde{x}_i = Ty_{i1}Ty_{i2}T \ldots Ty_{is_i}T$, $T = \underbrace{\beta \times \cdots \times \beta}_{p_1}$, $p_2 = p_1(|x_1|, \ldots, |x_s|, p(|x_1|, \ldots, |x_s|))$. T will be a counter recognizing the finish of the work of $\mathcal{B}$ on the list $x_1, \ldots x_{s+j}$; $j = 1, \ldots, M$.

Appropriate rules for $\mathcal{A}$ can be written as in section 2.

Step 2 *Input* : (12)

Output:

$$\alpha R_1\%R_2\% \ldots \%R_{s+M}\alpha, \tag{13}$$

where R_i is the result of the work of $\mathcal{B}$ on $x_1, \ldots, x_{s+j}$, $j = 1, \ldots, M$. The counters are contracted on 1 in any substitution of $\mathcal{A}$ in Step 2.

Step 3
Input : (13)
Output contains the letter γ iff $A(x_1, \dots x_s)$ is right.
Let a_n be the final letter for $\mathcal{B}$ if success. Rules for the Step 3 have the form:

$$\begin{aligned}
(x,y,z) &\longrightarrow Ł,\ \text{if } \%, a_n, \alpha \notin \{x,y,z\},\ \text{or } y=z=a_n,\\
(x,y,z) &\longrightarrow a_n,\ \text{if only one from } x,y,z \text{ is } a_n,\\
(\%, a_n, \%) &\longrightarrow Ł,\\
(a_n, \%, a_n) &\longrightarrow Ł,\\
(a_0, \alpha, \alpha) &\longrightarrow \gamma,\\
(\alpha, \%, a_n) &\longrightarrow Ł,\\
(a_n, \%, \alpha) &\longrightarrow Ł.
\end{aligned}$$

Inclusion (10) is proved.

Let's prove (11).
The difference from the previous case is only in Step 3. Rules have the form:

$$\begin{aligned}
(x,y,z) &\longrightarrow Ł,\ \text{if } a_n \notin \{x,y,z\},\ \text{or } y=z=a_n,\\
\text{in other cases } (x,y,z) &\longrightarrow a_n,\ \text{if } a_n \in \{x,y,z\},\\
(\alpha, a_n, \alpha) &\longrightarrow \gamma.
\end{aligned}$$

Theorem 3 is proved.

5. Resume

Thus, writing out sequentially the standard classes P, NP, HP and all the complexity classes defined above, we obtain the following chain

$$\mathrm{P} \overset{?}{\subseteq} \mathrm{NP} = \mathrm{PLSN} = \mathrm{NPLSN} \overset{?}{\subseteq} \mathrm{PH} \overset{?}{\subseteq} \mathrm{PLS} \overset{?}{\subseteq} \mathrm{NPLS} \neq \mathrm{P}_\alpha\,.$$

The following two problems: $\mathrm{P} \overset{?}{\subseteq} \mathrm{NP}$ and $\mathrm{NP} \overset{?}{\subseteq} \mathrm{PH}$ are the famous open problems in the theory of complexity. Therefore, in view of Theorem 3, Lindenmayer system (with the possibility of empty results) may be even more powerful tool of computations than nondeterministic Turing Machine.

Note that the nondeterminism of LS is conceptually classical. Harnessing of quantum mechanical effects for increasing the power of LS faces problems. The point is that reproductions of quantum states is forbidden by uncertainty principle.

7. References

1. C.H.Bennett, E.Bernstein, G.Brassard,, U.Vazirani, *Strengths and Weakness of Quantum Computing*, SIAM Journal of computing. (1997).
2. J.R.Koza, *Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming*, Symposium on Pattern Formation (SPF-93), Claremont, California, USA (1993).
3. J.Leeuwen van, *Formal models and Semantics Handbook of Theoretical Computer Science*, Elsevier Science Publishers B.V., Amsterdam, 1990.
4. A.Lindenmayer, *Mathematical models for cellular interactions in development 1. Filaments with one-sided inputs, 2. Filaments with two -sided input*, Jour. Theor. Biol. **jtb** (1968), 280-315.

5. A.Lindenmayer, *Developmental algorithms for multicellular organisms: a survey of L - systems*, Jour. Theor. Biol. **jtb** (1975), 3-22.
6. A.Lindenmayer, *Models for multicellular development: Characterization, inference and complexity of L-systems*, Lecture Notes in Computer Sience **281** (1987), 138 - 168.
7. G.Rozenberg, A.Salomaa (eds), *Lindenmayer systems: Impacts on theoretical computer sience, computer graphics and developmental biology*, Springer Verlag, Berlin, Heidelberg, New York, 1992.
8. W.J.Savitch, *Some characterizations of Lindenmayer systemsin terms of Cholmsky-type grammars and stack machines*, Information and control **27(1)** (1975), 37-60.
9. Paul M.B.Vitanyi, *Context sensitive table Lindenmayer systems and a relation to the LBA problem*, Information and control **33** (1977), 217 - 226.
10. S.Wolfram, *Cellular Automata and Complexity: Collected Papers*, Addison-Wesley, 1994.

Ozhigov Yuri., Department of Mathematics, Moscow State Technological University "Stankin", Vadkovsky per. 3a, 101472, Moscow, Russia

E-mail address: `y@ oz.msk.ru`